(12) UK Patent Application (19) GB (11) 2 391 419 (13) A

(43) Date of A Publication 04.02.2004

(21) Application No: 0213053.2

(22) Date of Filing: 07.06.2002

(51) INT CL[7]:
H04Q 3/00 // H04L 12/58 29/06

(52) UK CL (Edition W):
H4K KFMA KTN

(56) Documents Cited:
GB 2368163 A
GB 2315967 A
EP 0986229 A2
GB 2367714 A
GB 2001227 A
WO 1998/033333 A2

(58) Field of Search:
UK CL (Edition T) G4A AAP, H4K KFMA KTN
INT CL[7] G06F 1/00, H04L 12/28 12/58 29/06, H04Q 3/00
Other: Online: WPI, EPODOC, JAPIO

(71) Applicant(s):
Hewlett-Packard Company
(Incorporated in USA - Delaware)
3000 Hanover Street, Palo Alto,
California 94304, United States of America

(72) Inventor(s):
Matthew Murray Williamson
John Melvin Brawn

(74) Agent and/or Address for Service:
Hewlett-Packard Limited
Intellectual Property Section, Building 3,
Filton Road, Stoke Gifford, BRISTOL,
BS34 8QZ, United Kingdom

(54) Abstract Title: **Restricting the propagation of a virus within a network**

(57) Propagation of viruses from a given host within a network is reduced by preventing dispatch of data from the given host to other network hosts whose identities are not retained within a record of identities of recently contacted hosts.

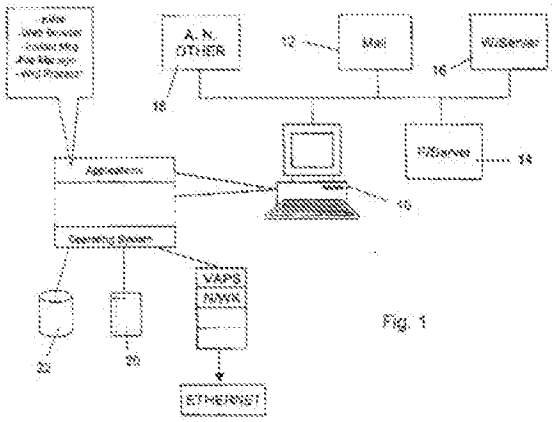


Fig. 1

GB 2 391 419 A

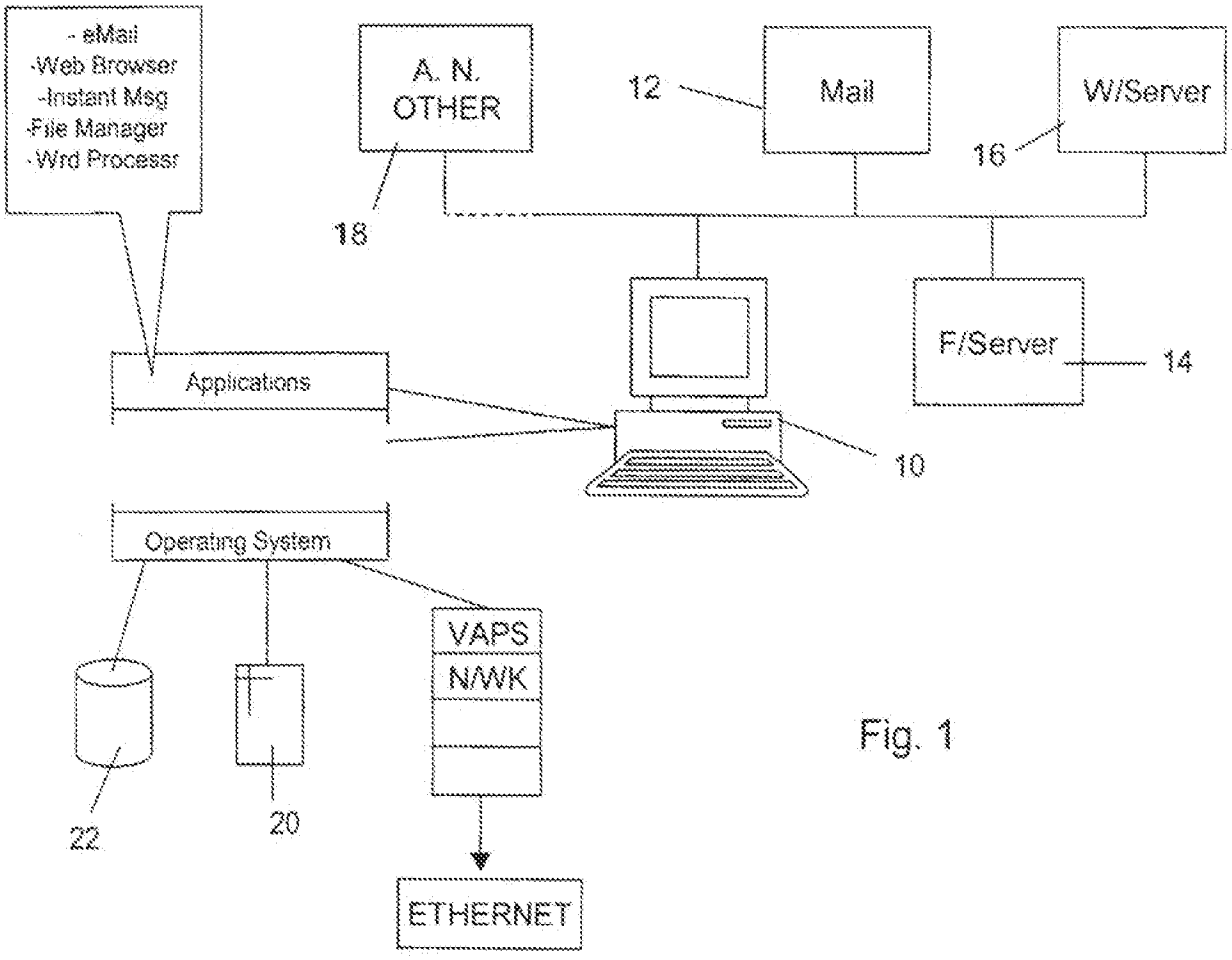
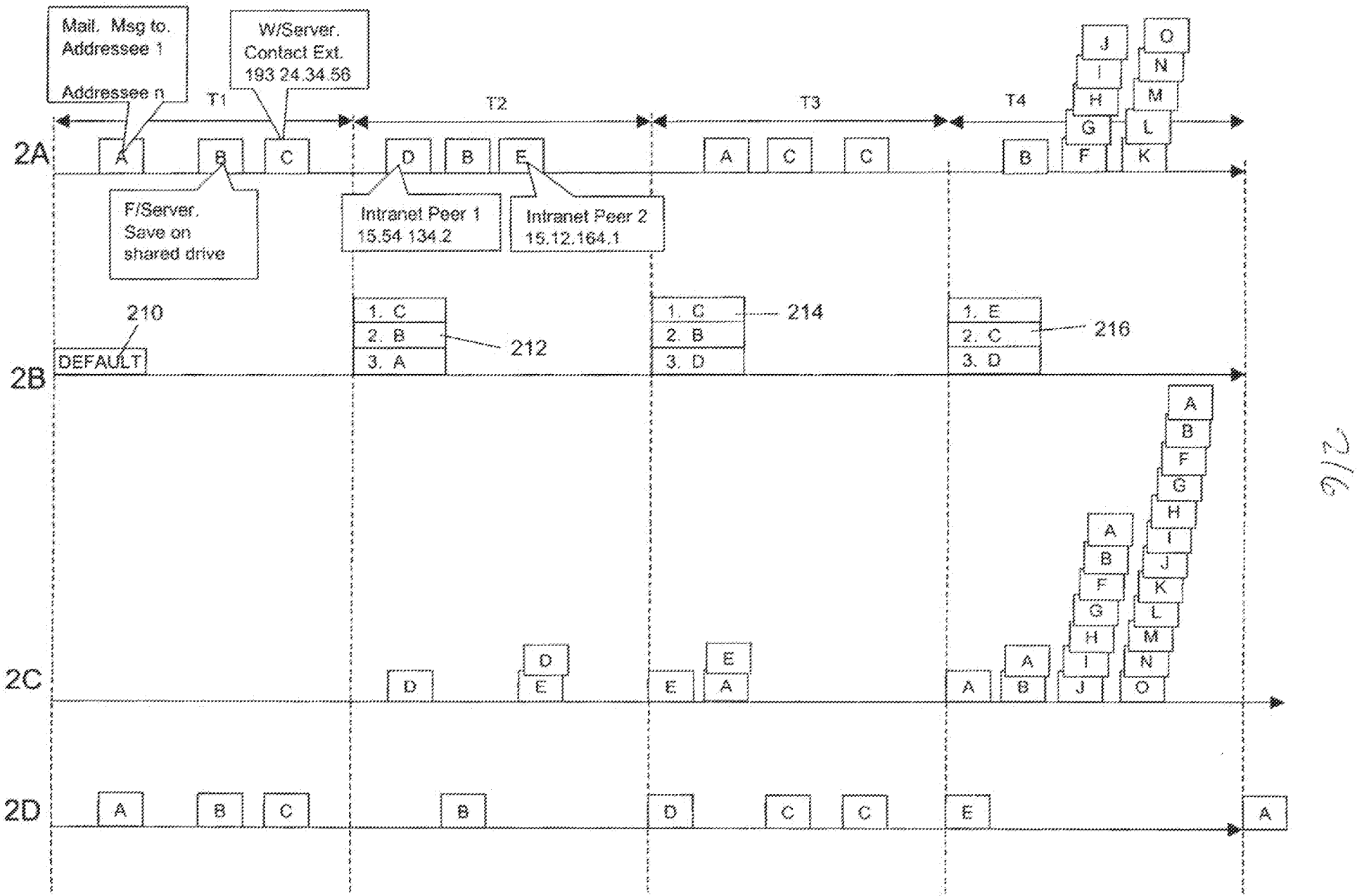

- eMail
-Web Browser
-Instant Msg
-File Manager
-Wrd Processr
A. N. OTHER
18
12
Mail
16
W/Server
F/Server
14
Applications
10
Operating System
VAPS
N/WK
22
20
ETHERNET

Fig. 1


Mail. Msg to.
Addressee 1
Addressee n
W/Server.
Contact Ext.
193 24.34.56
T1
T2
T3
T4
2A
2B
2C
2D
F/Server.
Save on
shared drive
Intranet Peer 1
15.54 134.2
Intranet Peer 2
15.12.164.1
210
DEFAULT
1. C
2. B
3. A
212
1. C
2. B
3. D
214
1. E
2. C
3. D
216

6/12

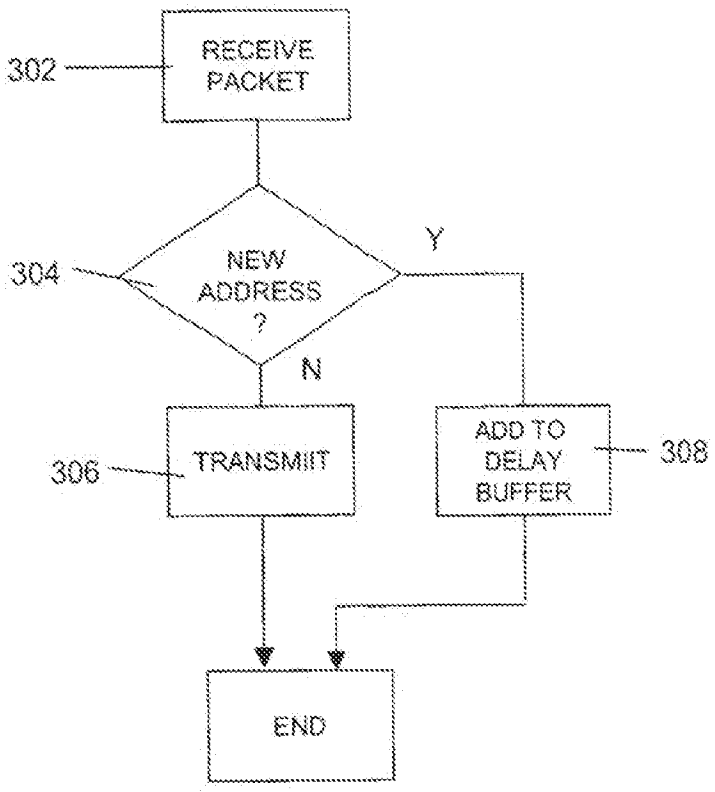

302
RECEIVE PACKET
304
NEW ADDRESS ?
Y
N
306
TRANSMIT
ADD TO DELAY BUFFER
308
END

Fig. 3

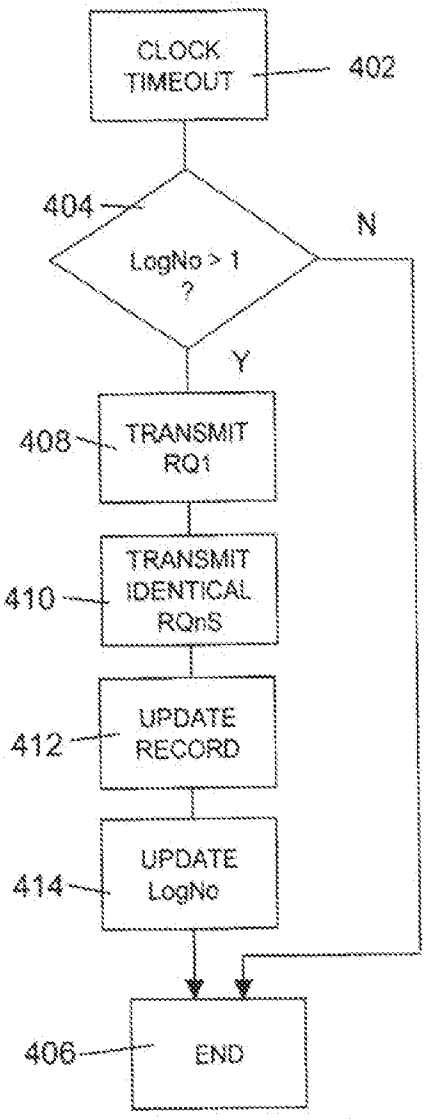

CLOCK TIMEOUT
402
404
LogNo > 1 ?
N
Y
408
TRANSMIT RQ1
410
TRANSMIT IDENTICAL RQnS
412
UPDATE RECORD
414
UPDATE LogNo
406
END

Fig. 4A

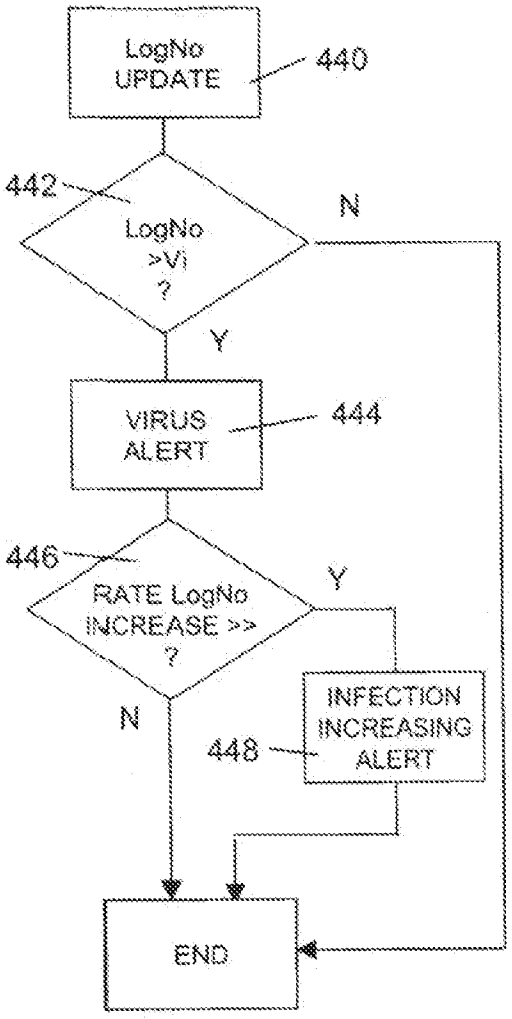

LogNo UPDATE
440
442
LogNo >Vi ?
N
Y
VIRUS ALERT
444
446
RATE LogNo INCREASE >> ?
Y
N
448
INFECTION INCREASING ALERT
END

Fig. 4B

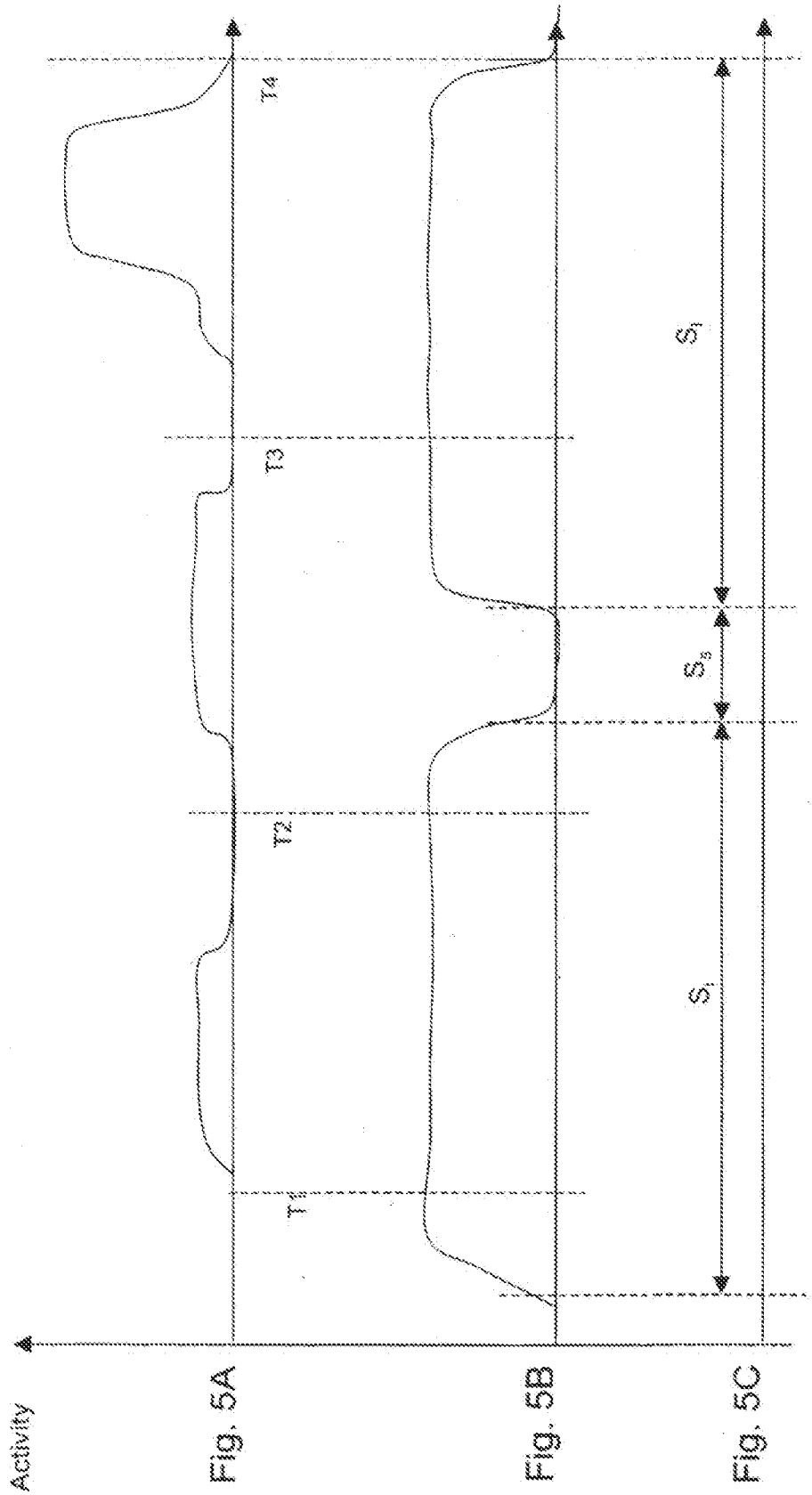


Fig. 5A

Fig. 5B

Fig. 5C

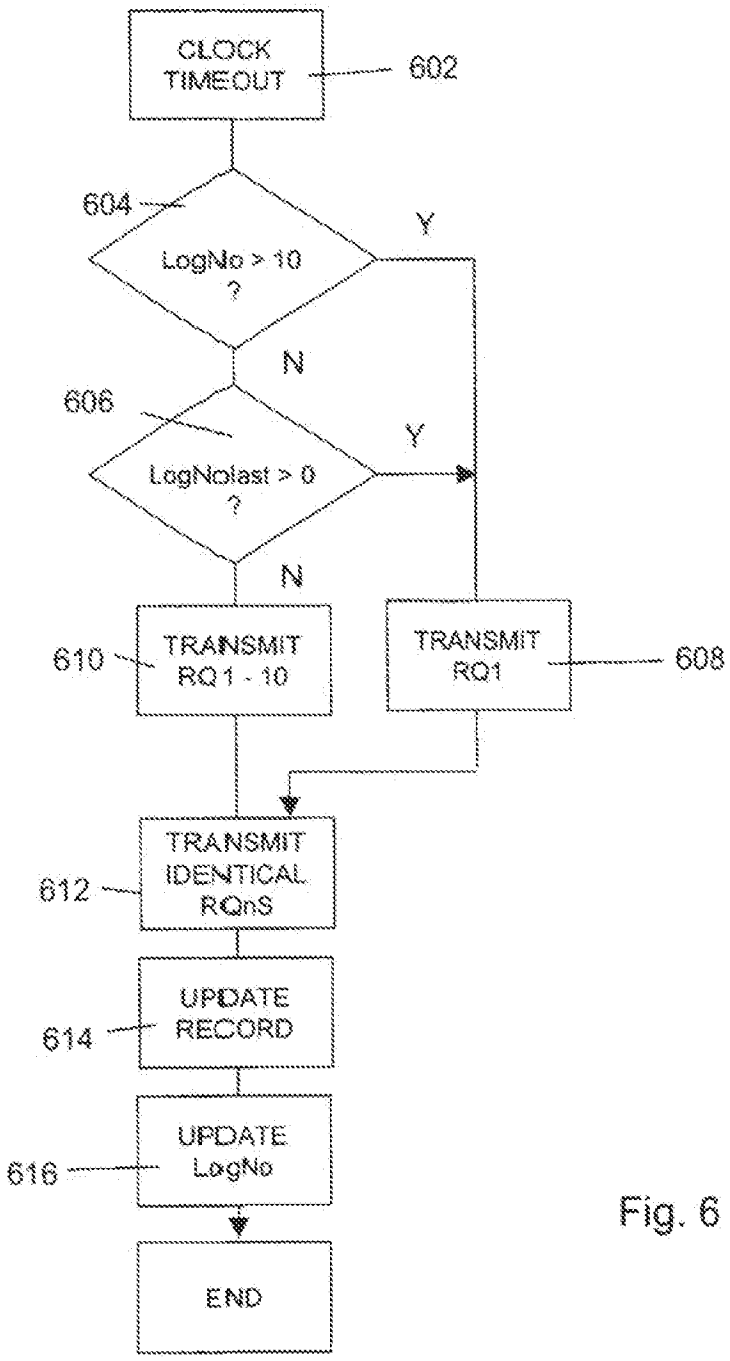

CLOCK TIMEOUT
602
604
LogNo > 10 ?
Y
N
606
LogNolast > 0 ?
Y
N
610
TRANSMIT RQ1 - 10
TRANSMIT RQ1
608
612
TRANSMIT IDENTICAL RQnS
614
UPDATE RECORD
616
UPDATE LogNo
END

Fig. 6

METHOD FOR RESTRICTING THE PROPAGATION OF A VIRUS WITHIN A NETWORK

The present invention relates to the propagation of viruses through a network of interconnected processing entities, and more particularly to reducing minimising or restricting such propagation.

In current network environments virtually any processing entity or "host", is at one time or another connected to one or more other hosts. Thus for example in the case of an IT environment, a host in the form of a computer (such as a client, a server, a router, or even a printer for example) is frequently connected to one or more other computers, whether within an intranet of a commercial organisation, or as part of the Internet. Alternatively, in the case of a communications technology environment, a host in the form of a mobile telephone is, merely by virtue of its intrinsic purpose, going to be connected to one or more other hosts from time to time, and an inevitable result is that the opportunities for the propagation of viruses are enhanced as a result. For example in the case of a computer virus known as the "Code Red" virus, once assimilated within a host the virus operates to generate Internet Protocol ("IP") addresses of other potential hosts at random, and then instructs the host to send a copy of the virus to each of these randomly-generated IP addresses. Although not all of the potential hosts are genuine (since the IP addresses are randomly generated), sufficient of the randomly generated addresses are real addresses of further hosts to enable the virus to self propagate rapidly through the Internet, and as a result to cause a substantial drop in performance of many commercial enterprise's computing infrastructure.

Within the context of this specification a virus is data which is assimilable by a host to cause a deleterious effect upon the performance of either: the aforesaid host; one or more other hosts; or a network of which any of the above-mentioned hosts are a part. Thus for example, a virus may act by becoming assimilated within a first host, and subsequent to its assimilation may then cause deleterious effects within that first host, such as corruption and/or deletion of files. In addition the virus may cause self-propagation to one or more further hosts at which it will then cause similar corruption/deletion and further self-propagation. Alternatively the virus may merely

be assimilated within the first host and cause no deleterious effects whatsoever, until it is propagated to one or more further hosts where it may then cause such deleterious effects, such as, for example, corruption and/or deletion of files. In yet a further alternative scenario, a virus may for example become assimilated within a first host, cause itself to be propagated to multiple other hosts within the network. The virus may have no deleterious effect upon any of the hosts by whom it is assimilated, however the self-propagation through the network *per se* may be of a sufficient magnitude to have a negative effect on the speed of "genuine" network traffic, so that the performance of the network is nonetheless affected in a deleterious manner. The three examples given above are intended for illustration of the breadth of the term virus, and are not intended to be regarded in any way as exclusively definitive.

It has been established that in situations where viruses are likely to cause deleterious effects upon either one or more hosts, or the network infrastructure as a whole, one of the most important parameters in attempting to limit and then to reverse such effects is the speed of propagation of a virus. Human responses to events are typically one or more orders of magnitude slower than the propagations speeds of viruses, and so substantial difficulties are frequently apt to arise within a network before any human network administrator is either aware of the problem, or capable of doing anything to remedy it. Therefore any reduction in the initial rate of propagation of a virus through a network is likely to be of benefit to attempts to limit any negative effects, and/or to remedy them.

According to a first aspect of the present invention, there is provided a method of restricting propagation of viruses in a network having a plurality of hosts, comprising the steps of:

monitoring network traffic from a first host of the plurality and establishing a record including at least data indicative of identities of hosts contacted by the first host; and

limiting network traffic from the first host over the course of a first time interval of predetermined duration, so that during the first time interval the first host is unable to contact more than a predetermined number of hosts not in the record.

By limiting the number of "new" hosts (i.e. hosts not present in the record) which may be contacted in any one time interval, the rate of propagation of a virus is limited to a predetermined level: only the predetermined number of new hosts can become infected per time window. Preferably the record of contacted hosts is adapted to take account of changes, and so the method preferably further comprises the step, where appropriate, of updating the record to include the identity of a host contacted during the first time interval and which was not in the record during the first time interval, this update record then being available for reference in a subsequent time interval.

Usually, in a network unaffected by viruses, the rate of connection to hosts not previously contacted is relatively low. The behaviour of a network affected by a virus however is different, in that typically a virus attempts to propagate rapidly through the network, and this propagation is manifested by connection, or attempted connection to a relatively large number of previously uncontacted hosts. Preferably therefore, one aspect of the present invention is that there is only a small effect on non-viral network traffic, but a significantly restrictive effect upon viral traffic.

Depending upon the nature of a virus, contact with other hosts within the network may be attempted directly, or via an intermediate or proxy host. Thus for example in the case of a virus which propagates via the medium of email, the virus may propagate by effectively emailing itself to one or more other hosts within the network, a typical example being a virus which causes the email programme within its host to generate an email containing the virus, and addressing this email to all contacts in the email programme's address book. Initially however, such an email message will have only a single destination host within the network: the mail server. In this instance therefore, although ostensibly only one other host within the network, i.e. the mail server is being directly contacted, the message to the mail server is carrying a plurality of further messages to addressees for whom, in the first instance the mail server is acting as a proxy. Preferably therefore, the use of proxy hosts is accounted for, and in one example this is achieved by determining the ultimate addressee or addressees of any outbound message (whether email or otherwise) and matching these against a suitable record prior to transmission.

Embodiments of the invention will now be described, by way of example, and with reference to the accompanying drawings, in which:

Fig. 1 is a schematic description illustration of an information technology network;

Figs. 2A-D are schematic illustrations of network traffic from a first host of the network illustrated in Fig. 1, and the management of such network traffic;

Fig. 3 is a flow chart illustrating operation of an aspect of a method according to the present invention;

Figs. 4A and B are flow charts illustrating the operation of further aspects of a method according to the present invention;

Figs. 5A-C illustrate a further embodiment of method according to the present invention; and

Fig. 6 is a flowchart of steps implementing the embodiment of method illustrated in Fig. 5C.

Referring now to Fig. 1, a network, which in the present illustrated example is an information technology network intranet, includes a plurality of interconnected hosts: a workstation 10 which is typically a personal computer for example, a mail server 12 ("Mail") which handles email communication within the network, a file server 14 ("F/Server") on which shared data within the network is stored, and a web proxy server 16 via which any communication between any host within the intranet and an external host passes. In addition the network includes further hosts not illustrated explicitly in Fig. 1, one of which 18 is illustrated under the denomination A. N. OTHER, and whose function within the network has no bearing upon the illustration of the present embodiment of the present invention.

The workstation 10 runs a plurality of software programmes concurrently; differing programmes operating at different levels within what may be thought of as a software hierarchy, or "stack". Low level software such as the operating system of the

workstation 10 is at the bottom of the stack, while other programmes, such as applications programmes interface with the operating system software rather than directly with the hardware of the workstation for the purpose of, for example managing the allocation of the workstation memory 20, and the saving and retrieval of files from storage 22 within the workstation etc.. Examples of the sort of applications programmes which run on the workstation 10 include programmes to handle the receipt and dispatch of email from the mail server 12, a web browsing programme, a file manager programme enabling the organisation and transportation of files, and instant messaging software enabling the dispatch and receipt of ASCII text messages directly to and from peers within the network. In addition the stack includes networking software which handles all communications passing into and out of the workstation 10 via the network link. In accordance with the illustrated embodiment of the present invention, a further software programme, Virus Anti-Propagation software (VAPS), runs within the stack adjacent the networking software.

The VAPS acts as a gateway for all outbound data from the workstation 10, and operates to restrict the propagation of viruses within the network by limiting the extent to which the workstation can engage in what may be thought of as "unusual" behaviour in contacting other hosts. In accordance with one aspect of the present invention, it has been established that in many networks, normal network traffic (i.e. non-virally related) is characterised by a relatively low rate of connection to hosts within the network which have previously not been contacted. In contrast, virally-related traffic is frequently characterised by a relatively high rate of connection, or attempted connection to previously uncontacted hosts. Broadly speaking, the function of the VAPS is to impede virally-related traffic, while allowing non-virally related traffic to flow with little or no impediment. In the present example the VAPS operates upon the basis of a series of time intervals or time windows, which in the present illustrated example are of predetermined and constant length T_s. In any given time window T_s the VAPS operates to prevent the host upon which it is running from contacting more than a predetermined number of "new" hosts, i.e. hosts whose identities differ from those specified in a record of identities of destination hosts most recently contacted. The record only holds a predetermined number N of destination host identities, so that a destination host is classified as new if it is not one of the N most recently contacted destination hosts. The number of new hosts allowed per time

window, and the value of N are determined on the basis of a policy, typically defined by a system administrator, and the policy is preferably formulated to take account of the nature of non virally-related network traffic. In this way, the VAPS operates to limit the speed at which a virus resident on the host may propagate from that host to other hosts within the network.

Referring to Fig. 2A, over the course of the time window T1, various applications programmes running on the workstation send requests to the VAPS to connect and send data to other hosts within the network ("outbound requests"): the email application programme, which requests dispatch of an email message (having multiple addressees) to the mail server 12, Mail (Request A), the file management application programme requesting dispatch of a file to the file server 14, F/Server in order to save a text document on a shared network drive (Request B), and the web browser programme which requests contact with the Web Proxy server 16, W/Server in order to contact a site external to the subnet within which the workstation 10 is located (Request C). In the present example, outbound requests to the VAPS from each of these hosts are each actually requests to provide a connection to an identified destination host, and the requests are transmitted via data packets from the relevant application programme. A request for connection, if allowed, is followed by data transmitted to the identified destination host. However, other forms of request are equally possible, depending upon the operation and/or configuration of the network, and the term "request" is intended to be interpreted broadly to encompass any indication that contact with an identified destination host is required, and this may for example be provided simply by the particular data packet intended for transmission to that host, rather than in the form of a packet requesting connection.

The these requests are processed in accordance with in incoming request routine, forming part of the VAPS (illustrated in Fig. 3), and the various steps that take place during the course of this routine will now be described in more detail with reference to the graphical representations of Figs. 2A-D in combination with the flowchart of Fig. 3. Subsequent to their generation by their respective applications programmes, each of the outbound requests, hereinafter abbreviated as Requests A, B, C passes through the stack from the respective application by which they were generated, to the VAPS, whereupon the process within the VAPS which processes the requests is initiated in step 302. Upon passing into the VAPS, the identity of the requested

destination host specified in each packet is matched with a dispatch record in which the identities of a predetermined number N (which in this example is 3) of destination hosts most recently contacted in the previous time window are stored (and which are shown for each time window in Fig. 2B), in order to determine whether the requested destination host is a new host, as represented at step 304. In the present example, somewhat artificially, but nonetheless serving to illustrate the principles underlying the present invention, the time interval T1 is the first time interval after start-up of the workstation 10. The VAPS therefore matches the destination host identities for each of the Requests A-C against identities held in a "default" dispatch record 210 for the time period T1, which may be (and in the illustrated example, is) simply a record of the three hosts most frequently contacted during the lifetime of the workstation. In the present example the three most frequently contacted hosts, and therefore the three identities retained in the default dispatch record are those of the mail server 12 (Request A), the file server 14 (Request B) and the web proxy server 16 (Request C). Since each of the three outbound requests from the workstation during the time period T1 identify a host destination matching one of the three host identities in the default dispatch record, and therefore non of the Requests is seeking to establish contact with a new destination host, the VAPS transmits each request at step 306, and in the present example this means that it allows a connection with each of these hosts to be established. Transmission of the request is illustrated schematically on the graph of Fig. 2D, which has the same time scale as Figs 2A_C, meaning that the temporal relationship between events illustrated in each of these graphs can be readily appreciated.

During the course of the second time interval T2, three further outbound requests identifying host destinations "Intranet Peer 1" (Request D), Request B (which as indicated above corresponds to the File Server 14) and "Intranet Peer 2" (Request E) are received by the VAPS from: an instant messaging application programme (in the case of Requests D and E), and the word processing application in the case of Request B. As in the previous time window, as each request passes to the VAPS, and as previously indicated in step 304, the identity of the host destination in the request is matched with the identities present in the dispatch record 212. The dispatch record however is now a genuine record of the identities of the three hosts contacted most recently during the previous time window T1 (although coincidentally this is identical

to the default dispatch record). Upon receipt of Request D, the VAPS establishes at step 304 that the identity of this host is not in the dispatch record, i.e. that it is a new destination host, whereupon the request is denied, and is instead stored in a delay buffer step 308. The delay buffer is effectively a queue of requests which have not been transmitted, and the contents of the delay buffer are illustrated schematically in Fig. 2C (the delay buffer is shown in Fig. 2C on each occasion that its contents change). It therefore follows that for each request illustrated in Fig. 2A, there is either a corresponding change in the delay buffer (illustrated in Fig. 2C) when the request is denied or transmission of the request (illustrated in Fig. 2D) when the request is transmitted. Request B is processed is previously indicated, and given that B is present in the dispatch record, this request is transmitted, which can be seen in Fig. 2D, while Request E, in a similar manner to that of the instance of Request D, is denied and added to the delay buffer, as illustrated in Fig. 2C.

Thus, at the end of the time period T2, no requests for connection to new destination hosts have been transmitted, and the delay buffer contains two new requests. At this juncture (i.e. at end of time period T2), the policy which the VAPS is designed to implement comes into play. In the present example, the policy provides that a single new host may be contact per time interval. This element of the policy is implemented by a first buffer management routine, which is illustrated schematically in flowchart form in Fig. 4A, and begins at step 402 with the advent of a clock timeout, that is to say that the clock (not shown) which defines the time intervals T_n has completed another time period. At step 404 the routine determines whether there are any requests in the delay buffer, and it does this using a variable known as LogNo, which is the number of requests in the delay buffer at any moment; if LogNo is not greater than 1, i.e. there are no requests in the delay buffer the routine ends at step 406. In the present illustrated example however it can be seen that over the course of the time interval T2 two requests, D and E have accumulated in the buffer, and so the routine proceeds to step 408, at which the first request RQ1 (i.e. the one which has been in the buffer for the longest time) is transmitted. At step 410, the routine then searches the buffer for other requests specifying the same destination host and transmits any such requests, the logic behind this being that, in the event there is a virus in the first transmitted request RQ1, further copies of the virus are not likely to be deleterious to any greater extent. This is followed at step 412 by updating the dispatch record so

that it accurately reflects the identity of the three most recently contacted hosts, and in Fig. 2B it can be seen that the dispatch record contains the identities D, C, B, which are the three most recently transmitted requests, as indicated in Fig. 2D. The final step in the first buffer management routine is the updating of the value of the variable LogNo denoting the size of the buffer, which in this example, following the transmission of the request D, is one (i.e. the single request E).

The buffer size plays an important role in implementation by the VAPS of another aspect of the policy, in that it is possible, if desired, to define a state of viral infection in terms of the size of the buffer, and the stage of any such viral infection by the rate of change of the buffer size. This follows from the generally different behaviour if virally-related and non virally-related network traffic, in that non virally-related or "legitimate" network traffic usually involves contacting only a relatively small number of new destination hosts, whereas, because viruses tend to propagate by transmission to as many disparate destination hosts as possible, an instance of a large number of requests to contact a new destination host will typically be indicative of viral infection. Given that the buffer is effectively a queue of new requests waiting to be transmitted, the size of the buffer is one indication of whether there is viral infection, since a large buffer size is indicative of a large number of requests to contact a new host within a short space of time. In addition, if the buffer size is increasing, this is correspondingly indicative of the onset of viral infection, whereas a steadily declining buffer size, although large, will be indicative of the end of a viral infection.

A second buffer management routine, illustrated in Fig. 4B implements this part of the policy, and is triggered at step 440 by the occurrence of an update of the value of LogNo (this being step 414 in the first buffer management routine), following which, at decision step 442, the routine determines whether the size of the buffer is greater than a quantity V_t, which the policy has determined represents viral infection, whereupon at step 444 it generates a virus alert. This may simply be a visual alert to a user of the workstation 10, or a message to the network administrator, or both, or even a trigger for automated action to shut the network down, as desired. At step 446, the routine determines whether the variable V_t is increasing above a given rate, and if it

is, issues a further warning indicating the onset of viral infection at step 448, following which the routine ends.

A situation in which the second buffer management routine generates a viral infection warning can be seen in Figs. 2A-D. During time interval T3, a single Request A (which it will be recalled from the time interval T1 is to contact the mail server), and two Requests C are received. Because the dispatch record 214 for this time interval does not contain Request A, this request is denied and sent to the delay buffer, while the two Requests C are transmitted. At the end of the time interval T3 the buffer therefore contains Request E (stored in the delay buffer since time interval T2) and Request A, and in accordance with the policy, the first buffer management routine transmits Request E at the end of the time interval T3, meaning that at the start of time interval T4 the buffer contains only Request A. The first Request for connection in time interval T4 is Request B (the File Server), which illustrates that over the course of three time intervals, during which only normal network traffic has been transmitted, connection has only been requested to five different destination hosts. However, Request B is nonetheless defined as new because it's not in the dispatch record 216 for time interval T4, and so is sent to the buffer (this action being illustrated at the same point in the timeline in Fig. 2C). After receipt of request B, two groups of five virtually simultaneous requests are received, F-J, and K-O, and since these are also new, they are also added to the buffer upon receipt and processing. Referring specifically to Fig. 2C during time interval T4, it can readily be seen that the buffer has increased from a size of one, to 12, and in accordance with the policy, this is defined as viral infection, since in the present example a buffer size of greater than five generates this alert. Moreover, size the rate of change is positive and rapid (from 1 to 12 in a single time interval), this is indicative of the onset of infection.

In the example described above the VAPS has been configured to delay outbound requests, and as seen this has the advantage of being able to use the delay buffer to provide useful information. In addition, delaying outbound requests for connection is generally regarded as being compatible with the operation of many computer systems and networks. However, the VAPS may be configured to operate in a number of ways. For example, in accordance with an alternative embodiment, where the

computer system permits, the VAPS may, having denied the request for connection, and simply return a suitable error message to the dispatching application programme by which the packet was generated, and then delete the packet. In accordance with this embodiment the dispatching application programme must, if the packet is eventually to be successfully dispatched then resend the packet the VAPS. In this alternative embodiment, the policy relating to the number of new requests which are to be transmitted per interval may be implemented by initialising a variable corresponding to the number of new requests received in a particular time interval, and augmenting this variable whenever a new request is received. Requests may then either be instantaneously transmitted (in the same manner as requests already in the dispatch record) or denied and deleted on the basis of whether the variable indicative of the number of new requests per time interval has reached a maximum set in accordance with the policy (i.e. in the previous example, one).

In the present example, the dispatch record lists transmitted requests in historical order, with the ordinal numbering signifying the temporal order in which the hosts where contacted, i.e. No. 1 indicating the host most recently contacted, and No. 3 indicating the host contacted the longest time previously (or "first in first out". This is not essential, and it is equally possible to list the transmitted requests in another order, such as "first in last out" for example.

In the scenario illustrated in connection with Fig. 2, a single outbound request (Request A) to the VAPS, specifying a single destination host, namely the mail server, actually contains a plurality of email messages to different specified addressees. This outbound request may therefore be thought of as a carrier request for a plurality of sub-requests, here having the form of putative email messages intended for dispatch from the mail server to a list of addressees specified within the outbound carrier request (similarly, the mail server may be thought of as acting as a proxy destination host for the ultimate addressees specified in the outbound carrier request). In this situation, allowing transmission of the data packet constituting the message to the mail server will in fact effectively allow the workstation to contact multiple other hosts within the network (i.e. the specified addressees) all of which may be new, even though, in accordance with the routine described in connection with Figs. 3, the outbound carrier request will only count as a single request which may not even be

recognised as new if, as may be likely, the mail server is identified in the current dispatch record. In such a situation therefore, if the VAPS operates simply to restrict the number of new destination hosts to be contacted per time window on the basis only of those destination hosts which are ostensibly identified in the outbound request, the desired restrictive effect on virus propagation may be circumvented or reduced, because a single outbound request specifying the mail server does not necessarily represent only a single email subsequently propagating through the network after processing and forwarding by the mail server.

In a modification of the embodiment thus far described therefore, the VAPS includes within its routine a step of identifying the application programme by which an outbound request has been generated. Because certain applications programmes are more likely than others to use outbound carrier requests which invoke the use of a proxy (for example the above-mentioned instance of email, or the case of a web browser programme) it is possible in advance to specify criteria, based on the provenance of an outbound request, identifying those outbound requests likely to be carrier requests. If the packet is generated by one such specified application programme, then the VAPS invokes the use of the application programme concerned to reveal the identities of the destination hosts specified in the sub-requests; here the eventual addressees for whom the email message is intended. Once the identities of the genuine or ultimate addressees have been obtained, there are several options for processing the request. In accordance with one alternative the identities of the destination hosts specified in the sub-request can be regulated in accordance with the same policy which applies to all other requests for connections, and they can be matched against the host identities within the dispatch record in the manner previously described in the embodiment of Figs. 3. In the event that the message contains more new addressees than the policy which the VAPS is implementing will allow to be transmitted in a single time window, then what may be thought of as the surplus addressees may, depending upon the operation of the email programme, either be purged from the list, and the message transmitted (such surplus messages may alternatively be dealt with an a different manner, which may also be specified in accordance with the policy), or they may be stored in a delay buffer as illustrated in connection with Figs. 2 and 3.

Since in the case for example of email, the use of outbound carrier requests to a host acting as a proxy for the ultimate addressees of the email messages is the norm, it is, in a modification, possible for different versions of VAPS to run simultaneously, effectively operating in parallel with each other: one which applies to hosts specified in the outbound request (including carrier requests), and another which applies to hosts specified in any sub-requests identified by the email application programme. In such a situation, each VAPS will operate independently, using its own dispatch record, and implementing a policy for outbound requests tailored to the traffic it is set up to control, for example in the manner previously described and illustrated in connection with Figs. 2 and 3. The two policies may be the same (e.g. a dispatch record of 3 identities, a time window of constant duration T_n, and one new host per outbound request/sub-request), or different as desired.

The choice of the length of the time window, the number of identities retained in a dispatch record, and the number of new hosts to be allowed per time window are all dependent upon the likely "normal" performance of the network within which the VAPS is operating, and more particularly, the nature of the network traffic the VAPS is intended to control. Therefore, while a policy such as that illustrated in connection with Figs. 2 and 3 may be effective in limiting the propagation of viruses through the network to a rate of infection of one new host per time interval, it may also be susceptible to interfering with non virally-related, or "legitimate" network traffic whose characteristic behaviour differs substantially from the policy the VAPS is implementing. To ameliorate this difficulty, it is possible to provide a version of VAPS for each application programme from which network traffic emanates, with each VAPS implementing a policy tailored specifically to minimise the level of impediment to legitimate network traffic.

Referring now to Fig. 5A, a plot of activity (i.e. the number of requests processed by the VAPS) against time is illustrated for example of Fig. 2A. From this graph it can be readily appreciated that prior to the viral infection signified by the rapid increase in the number of requests during the time interval T4, only a relatively low number of requests are processed per time interval, and that therefore it is possible to use the VAPS to implement a policy preventing connection to more than one new host per

time interval without impeding legitimate network traffic to any significant extent. Consider however an excerpt of a graph illustrating legitimate traffic flow in Fig. 5B, where there are significant levels of activity, interspersed by a much shorter period of time during which there is no activity at all. Applying the rather simple policy of permitting connection to one new host per time interval, where all time intervals are of the same duration would significantly impede the flow of the legitimate network traffic illustrated in Fig. 5B. Ideally therefore, an alternative policy is required which accounts for the nature of this legitimate traffic flow. An example of such a policy is illustrated referring now to Fig. 5C, where two sorts of time intervals are illustrated: S_l, a relatively long time interval, and S_s, a relatively short time interval. From Fig. 5C it can be seen that when placed together alternately, the time intervals S_l corresponds to the time interval in the graph of the traffic flow from Fig. 5B where there is a flow of traffic, and the time interval S_s to the time interval between two such time intervals, where there is no traffic flow. By segmenting time for a VAPS using these two time intervals therefore, it is possible to construct a policy which matches closely the legitimate behaviour illustrated in Fig. 5B, but still provides an impediment to the propagation of viruses. Such a policy for the VAPS may be implemented using the variable LogNo, which as explained above corresponds to the number of requests present in the delay buffer at the end of any given time interval. In the present example it is desirable to implement a policy which does not impede the free flow of the legitimate traffic pattern illustrated in Fig. 5C, and referring now to Fig. 6, to this end a modified first buffer management routine is provided. Following a clock timeout at step 602, the routine determines at step 604 whether the LogNo is greater than a predetermined number, in this instance 10, this number being chosen, in conjunction with the number of request identities held in the dispatch record, to be equal or slightly larger than the number of requests typically received during a "long" time interval S_l. If LogNo is greater than this number, then the routine defaults to step 608, where it transmits only the first request in the delay buffer, and then proceeds to steps 612 to 616 where identical requests are transmitted the record is updated, and the value of LogNo is updated. If LogNo is less than 10, i.e. less than 10 new requests have been received during the course of that time interval, then the routine proceeds to step 606, at which it determines whether a further variable LogNoLast, equal to the number of new requests received during the previous time interval, is greater than zero. If it is, then the routine defaults once

again to step 608 where only a single request is transmitted from the delay buffer. If it is not, i.e. no new requests were received during the previous time interval, then the routine acts to transmit, at step 610, requests 1-10 from the delay buffer, followed by the steps 612 to 616. Thus, when 10 or less new requests are received during a time interval, and no new requests were received during the previous time window, the routine operates to transmit all 10 requests. This mimics the legitimate activity during a "long" time interval S_l, where the activity level is relatively high, but in the previous short time interval activity was zero. Correspondingly, in any time window where there were more than 10 new requests (i.e. a greater level of activity than usual in a longer time interval) or where, in the previous time window there were more than zero new requests (which is the pattern of legitimate traffic flow illustrated in Fig. 5B), the routine defaults to what may be thought of as the "standard" policy of one new request per time interval, thus throttling activity differing from usual legitimate activity, and which is likely to be virally-related. The modified routine thus implements a policy which conforms generally to the behaviour pattern illustrated in Fig. 5C.

This modified policy implementation has been achieved using two time intervals of different lengths, and a modified version of the buffer management routine, effectively to augment the number of destination hosts which, ultimately (i.e. in this example, at the end of time intervals S_l) end up not being regarded as new. It is however possible to implement policies by varying other parameters, such as the number of destination host identities retained in the dispatch record, thereby increasing for any given time interval, the number of destination hosts which will not be regarded as being new, and consequently transmitting a greater number of destination hosts per time interval (or in the case of Fig. 5C and 6, per alternate time interval). This would be appropriate in circumstances where the legitimate traffic flow of Fig. 5B was characterised by contact with 10 destination hosts whose identities are the same, or similar each time. To achieve this for the traffic flow of Fig. 5B, two dispatch records for the destination hosts are used: one for the time intervals S_l, containing 10 destination host identities, and the other for the time intervals S_s, containing no destination host identities, with the two dispatch records being used alternately. However, as indicated above, where the legitimate traffic flow is characterised by contact with (in this example) 10 *different* destination hosts each

time interval S_t, this modification would not be appropriate because it would still impede this legitimate traffic flow.

In yet a further and more refined version of this policy implementation, in which provision is made for contact with 10 new destination hosts per time interval S_t, a modified version of the routine of Fig. 3, in which the further variables NreqNo, and NreqNolast, denoting the number of new requests in a particular time interval, and the number of new requests the preceding time interval (and thus the real time equivalents to LogNo and LogNolast) are used to transmit new requests contemporaneously, up to a maximum of 10 per time interval, provided that the two criteria of steps 604 and 606 are satisfied, i.e. that ReqNo is less than 10, AND ReqNolast was equal to zero. This modification has the advantage of allowing requests to pass immediately, which in cases where legitimate traffic levels are high, prevents undue impediment to traffic flow. In this modified version new requests which are not transmitted are once again stored in the delay buffer, which as previously, *inter alia* enables an indication of viral infection from the value of the LogNo variable.

The operation of the VAPS has been illustrated herein on a single workstation within a network. However, in order to be most advantageous it is desirably implemented on a plurality of hosts within the network; the greater the number of hosts upon which it is implemented resulting in a greater limit on the ability of viruses to propagate through the network.

The use of a different running concurrently, with one VAPS per application programme is a preferred embodiment of the present invention, since it enables the implementation of different policies for different application programmes and thus policies designed to minimise impediment to legitimate traffic flow, while simultaneously providing protection against viral propagation via the appropriated use of application programmes. Other implementations are possible, such as: a single VAPS implementing a single policy for all applications programmes; a plurality of VAPS, some of which deal with traffic from a specified application programme, and some of which deal with traffic to a particular destination port (which may be thought of generally as dealing with traffic using a particular communications protocol); or a

plurality of VAPS may be provided with each one dealing with traffic for a particular destination port.

5

CLAIMS

1. A method of restricting propagation of viruses in a network having a plurality of hosts, comprising the steps of:

monitoring network traffic from a first host of the plurality and establishing a record which is at least indicative of identities of hosts to whom data has been sent by the first host; and

limiting passage of data from the first host to other hosts within the network over the course of a first time interval, so that during the first time interval the first host is unable to send data to more than a predetermined number of hosts not in the record.

2. A method according to claim 1 wherein the record is established by monitoring the identities of hosts to whom data has been sent by the first host over the course of a further time interval which precedes the first time interval.

3. A method according to claim 2 further comprising the step of updating the record to include the identity at least one host to whom data was sent by the first host during the first time interval and which was not in the record during the first time interval.

4. A method according to claim 3 wherein the record contains a predetermined number of identities, and upon updating the record with the identity of a host not previously in the record, the identity of a host previously stored in the record is deleted therefrom.

5. A method according to claim 2 wherein the first and further time intervals are of predetermined duration.

6. A method according to claim 5 wherein the first and further time intervals are of equal duration.

7. A method according to claim 1, wherein the method includes the step of generating a request to send data from the first host to another host within the

network, and wherein identities of hosts to whom data has been sent are monitored by monitoring requests.

8. A method according to claim 7 further comprising the step of establishing whether a request contains at least one sub-request to send further data to a further host on whose behalf a destination host specified in the request acts as a proxy for receipt of the further data.

9. A method according to claim 8, wherein a further record is established of other hosts within the network whose identity is specified in a sub-request, the method further comprising the step of limiting passage of further data from the first host to other hosts within the network over the course of the first time interval, so that during the first time interval the first host is unable to send further data to more than a predetermined number of hosts not in the further record.

10. A method according to claim 1 further comprising the step of diverting requests to contact hosts not in the record to a delay buffer.

11. A method according to claim 10 further comprising the step, at the end of the first time interval, of transmitting the predetermined number of requests from the delay buffer.

12. A method according to claim 11 further comprising the step of monitoring the size of the delay buffer, and in the event that the delay buffer exceeds a predetermined size, generating a warning.

13. A network having a plurality of hosts, at least a first host of which includes a gateway for outbound data, the gateway being adapted to:

monitor requests from within the first host to send data to other hosts;

maintain a record of identities of hosts to whom data has been sent; and

prevent dispatch of data from the first host to another host within the network whose identity is not in the record.

14. A network according to claim 13 wherein the gateway is adapted to monitor requests from the first host to transmit data to another host within the network over the course of a series of time intervals, and is additionally adapted to transmit data to another host within the network whose identity is not in the record on a predetermined number of occasions in any given time interval.

15. A network according to claim 14 wherein the gateway is adapted to divert requests to contact hosts whose identity is not in the record to a delay buffer.

Application No: GB 0213053.2 **Examiner:** Daniel Voisey
Claims searched: 1 to 15 **Date of search:** 31 October 2002

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.T): H4K (KFMA, KTN); G4A (AAP)

Int Cl (Ed.7): H04L 12/26, 12/58, 29/06; H04Q 3/00; G06F 1/00

Other: Online: WPI, EPODOC, JAPIO

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A	GB 2368163 A	(IBM) see abstract.	
A	GB 2367714 A	(MESSAGELABS) see abstract.	
A	GB 2315967 A	(3COM) see particularly page 2 line 7 to page 3 line 17 and figure 2.	
A	GB 2001227 A	(INTERNATIONAL STANDARD ELECTRIC) see abstract.	
A	EP 0986229 A2	(JSB) see particularly column 2 lines 11 to 23, column 5 lines 8 to 37.	
A	WO 98/33333 A2	(BT) see abstract.	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category	P	Document published on or after the declared priority date but before the filing date of this invention
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application